The Bell System Technical Journal
Vol. 51, No. 1, January, 1972
Printed in U.S.A.

Optimal Test Point Selection for Sequential Manufacturing Processes

By M. R. GAREY

(Manuscript received June 23, 1971)

Consider a manufacturing process, such as the production of complex semiconductor devices, which consists of the sequential application of n possibly unreliable operations, t_1, t_2, $\cdots$, t_n. Let c_i be the cost incurred in performing operation t_i, and let p_i be the probability that t_i will be performed successfully. Clearly one would prefer to reject immediately any item as soon as a faulty operation has been performed upon it in order to avoid the unnecessary cost of further processing that item. For this purpose, we shall assume that, immediately following each operation t_i, it is possible to apply a perfectly reliable test T_i, for determining whether or not the item should be rejected at that point, where the cost incurred by applying T_i depends only upon the point i of test application and the last previous point at which such a test was applied. Since the application of tests entails additional costs, careful analysis is required to determine which tests are sufficiently useful to justify that additional cost. Using a dynamic programming approach, we derive a useful and efficient algorithm which utilizes the test and operation costs, along with the operation failure probabilities, to determine a set of testing points which will result in the minimal expected manufacturing cost. We then show how it is possible to further improve upon our algorithm for the particular case in which all test costs depend only upon the point of test application.

Certain types of products, such as many semiconductor devices, are manufactured by performing a linear sequence of individual operations, where many of the individual operations have a nonnegligible probability of yielding an undesired result. In such cases it would be preferable to immediately reject a faulty item to avoid the unnecessary, and useless, cost of further processing that item. However, the introduction of tests into the manufacturing process, for determining

whether or not an item should be rejected, will itself introduce additional costs in such a way that the total expected cost of the manufacturing process may increase. In addition, certain choices of test points, though reducing the total expected cost, may be considerably more costly than certain other choices of test points. We shall present an efficient and useful algorithm that uses the various test and operation costs, along with the operation failure probabilities, to determine an optimal set of testing points, which minimizes the expected cost of the entire manufacturing process.

Consider a manufacturing process consisting of n individual operations, or tasks, t_1 , t_2 , $\cdots$, t_n , which are to be performed sequentially in the fixed order of their indices. Associated with each task t_i are the cost $c_i > 0$ incurred by performing t_i and the success probability p_i , $0 < p_i \leqq 1$, that operation t_i will have the proper result, assuming all previous operations were successful. Immediately after any operation t_i it is possible to perform a perfectly reliable test T_i which determines whether or not all previous operations, including t_i , have been successful, that is, whether or not the item should be rejected at that point. The cost for performing test T_i is given by $C_i > 0$. We justify this simple testing cost by noting that in many manufacturing procedures of this type the major determinant of testing cost is merely the cost of removing the item from the assembly line and preparing it for test application. However, we shall later generalize the testing cost to allow a dependence upon the previously untested operations. We now develop an algorithm which uses the knowledge of these costs and probabilities to determine a choice of test points (a subset of the T_i) which results in the lowest possible expected manufacturing cost per item processed. This is equivalent to minimizing the expected cost per fault-free item produced, because the introduction of tests into the process does not change the probability, given by $\prod_{i=1}^{n} p_i$, that an item is successfully processed by all n tasks. The tests merely serve to reject already faulty items to avoid the cost of useless additional processing.

We solve this problem by the method of dynamic programming.[1] Following each operation t_i it is necessary to decide whether or not test T_i should be applied. To make this decision we need only compute the minimal expected costs for completing the process assuming that T_i is or is not utilized and choose the smaller. The computations of these two expected costs use the cost of T_i , the probability p that the item is not faulty at this point, and the cost of t_{i+1} . For the case that T_i is used, we also need the minimal expected cost for completing the

process under the assumption that the item is fault-free before t_{j+1} is performed. For the case that T_j is not used, we need the minimal expected cost for completing the process under the assumption that the item is not faulty with probability p before t_{j+1} is performed.

Formally, let a state of the process be described by $[j, p]$, with $1 \leqq j \leqq n$ and $0 < p \leqq 1$, where the process is in state $[j, p]$ if and only if operation t_j has just been applied and the resulting item is fault-free with probability p. There is also a "reject" state R entered when a test rejects a faulty item. The initial state of the process is $[1, p_1]$, and the final states are all those states of the form $[n, p]$, $0 < p \leqq 1$, plus the reject state R. The transitions between states are determined by the decisions on whether or not to apply a test before the next operation. If the process is in state $[j, p]$ and test T_j is not applied, the next state will be $[j + 1, p \cdot p_{j+1}]$. If the process is in state $[j, p]$ and test T_j is applied, the next state will be either $[j + 1, p_{j+1}]$, with probability p, or R, with probability $1 - p$. The problem is to find a sequence of states to transform the process from the initial state to any final state, with minimal expected total process cost.

Let $K[j, p]$ be the minimum expected cost to get from state $[j, p]$ to a final state and let $S[j, p]$ be the corresponding optimal set of test points T_k. We then have

$$K[j, p] = \min \begin{cases} C_j + p \cdot (c_{j+1} + K[j + 1, p_{j+1}]) \\ c_{j+1} + K[j + 1, p \cdot p_{j+1}] \end{cases}$$

and $S[j, p]$ equals either $S[j + 1, p_{j+1}] \cup \{T_j\}$ or $S[j + 1, p \cdot p_{j+1}]$, depending upon whether the first or second expression is smaller. For a final state $[n, p]$, $0 < p \leqq 1$, we have $K[n, p] = 0$ and $S[n, p]$ empty. (In our basic model, the final test will never be applied, since it can only increase the total expected cost. However, the results can also be applied if test T_n is always required as a final product quality check, in which case $K[n, p] = C_n$ and $S[n, p] = \{T_n\}$.)

The difficulty with using these equations to obtain an optimal solution is that, since p can take any value from 0 to 1, there are an infinite number of equations to solve. Fortunately, for states of the form $[j, p]$, p can take only a finite number of possible values, $\prod_{k=1}^{j} p_k$, $\prod_{k=2}^{j} p_k$, $\cdots$, $\prod_{k=j}^{j} p_k$, depending upon the last test which had been applied. Thus, if none of the tests T_i, T_{i+1}, $\cdots$, T_{j-1} were applied, and either $i = 1$ or T_{i-1} was applied, then $p = \prod_{k=i}^{j} p_k$. Accordingly we shall now describe a state of the process by (j, i), $i \leqq j$, where the process is in state (j, i) if and only if

(*i*) operation t_j has just been applied,
(*ii*) none of the tests T_i, T_{i+1}, $\cdots$, T_{j-1} were applied, and
(*iii*) either $i = 1$ or T_{i-1} was applied.

The initial state is then state (1, 1) and the final states are of the form (n, i), $1 \leqq i \leqq n$, plus the reject state R. If the process is in state (j, i), the next state will be either $(j + 1, j + 1)$, with probability p, or R, with probability $1 - p$, if T_j is applied or $(j + 1, i)$ if T_j is not applied. With $K(j, i)$ and $S(j, i)$ defined in the same way as previously, the dynamic programming equations become

$$K(j, i) = \min \begin{cases} C_j + \left(\prod_{k=i}^{j} p_k\right) \cdot (c_{j+1} + K(j + 1, j + 1)) \\ c_{j+1} + K(j + 1, i) \end{cases}$$

and $S(j, i)$ equals either $S(j + 1, j + 1) \cup \{T_j\}$ or $S(j + 1, i)$, depending upon whether the first or second expression is smaller. We also have $K(n, i) = 0$ and $S(n, i)$ empty for $1 \leqq i \leqq n$. This set of equations can be solved iteratively to determine the optimal solution.

Observe now that no additional complications arise if we generalize the test costs to allow a dependence upon the state (j, i). Accordingly we redefine the cost of test T_j, when the previously untested operations are t_i, t_{i+1}, $\cdots$, t_j, to be given by $C_{j,i} > 0$, allowing the additional dependence upon i. We can then rewrite our dynamic programming equations as

$$K(j, i) = \min \begin{cases} C_{j,i} + \left(\prod_{k=i}^{j} p_k\right) \cdot (c_{j+1} + K(j + 1, j + 1)) \\ c_{j+1} + K(j + 1, i) \end{cases}$$

with $K(n, i) = 0$ for $1 \leqq i \leqq n$, with $S(j, i)$ corresponding as before. The following algorithm uses these equations to iteratively solve for the optimal set of test points.

Algorithm 1. *Computation of Optimal Testing Points*

(a) For each i, $1 \leqq i \leqq n$, set
$K(n, i) \leftarrow 0$ and $S(n, i) \leftarrow \varnothing$ (empty set).
(b) Set $j \leftarrow n - 1$.
(c) Set $i \leftarrow j$ and $P \leftarrow 1$.
(d) Set $P \leftarrow P \cdot p_i$.
(e) Compute

$$K(j, i) = \min \begin{cases} C_{j,i} + P \cdot (c_{j+1} + K(j + 1, j + 1)) \\ c_{j+1} + K(j + 1, i) \end{cases}$$

and set $S(j, i)$ to $S(j + 1, j + 1) \cup \{T_i\}$ or $S(j + 1, i)$, depending, respectively, on whether the first or second expression is smaller.

(f) If $i > 1$, set $i \leftarrow i - 1$ and go to (d).

(g) If $j > 1$, set $j \leftarrow j - 1$ and go to (c).

(h) The optimal set of test points is given by $S(1, 1)$ and yields a manufacturing process with total expected cost per item processed of $c_1 + K(1, 1)$.

Notice that it is not necessary to actually construct and save all the optimal partial sets $S(j, i)$ but merely to note whether or not test T_i is included in $S(j, i)$. The optimal test set can be constructed directly from this information upon completion of the algorithm. The number of operations performed by the algorithm is at most proportional to the square of the number of tasks in the manufacturing procedure under analysis. Thus the algorithm can be economically used for extremely large problems when programmed for a digital computer, and many problems can be reasonably solved by hand calculation. The details of the algorithm are illustrated with a small example in Appendix A.

Notice that if physical constraints prevent the application of a test following certain tasks, this can be handled in the algorithm merely by automatically setting $K(j, i)$ to $c_{i+1} + K(j + 1, i)$ whenever T_i is not allowed. This is equivalent to combining tasks t_i and t_{i+1} into a single task having cost $c_i + c_{i+1}$ and success probability $p_i \cdot p_{i+1}$.

We now return briefly to the original case where the cost of test T_i depends only upon j and show how Algorithm 1 can be improved for this case. We let

$$D(j, i) = C_i + \left(\prod_{k=i}^{j} p_k\right)(c_{j+1} + K(j + 1, j + 1)) - c_{j+1} - K(j + 1, i).$$

Thus $D(j, i)$ is merely the difference of the two expressions which must be compared in order to determine whether or not to apply test T_i when in state (j, i). An optimal solution is formed by choosing to apply test T_i from state (j, i) if and only if $D(j, i) < 0$. We can then prove the following theorem.

Theorem 1: For all j, $1 < j < n$, and all i, $1 < i < j$, $D(j, i) \geqq D(j, i - 1)$.

The proof of Theorem 1 is given in Appendix B. Simply stated, Theorem 1 tells us that, for fixed j, the difference function $D(j, i)$ changes sign only once. In particular, it implies that if test T_i is to be applied in state (j, i), then test T_i should also be applied in each of the

states $(j, i-1)$, $(j, i-2)$, $\cdots$, $(j, 1)$. This enables us to modify step (e) of Algorithm 1, when all test costs depend only on j, to omit the computation of $c_{j+1} + K(j+1, i)$ in determining $K(j, i)$ whenever $T_j \in S(j, i+1)$. In fact, for very large problems, one might consider using a binary search, for each j, to discover the value $i = i_0$ at which $D(j, i)$ changes sign. Once that point has been found, each $K(j, i)$ can be computed by evaluating only one of the two expressions in the minimization, the proper one being determined by whether i is less than i_0 or greater than i_0. Though the number of operations required will still increase proportionally to n^2, this method will substantially reduce the actual number of operations performed.

A related problem with tests which merely test for the success or failure of a single operation, all of which must be applied, but which are not restricted as to point of application in the process sequence is solved as a special case in Ref. 2. The algorithm given there requires time bounded by the square of the number of tasks in the production sequence. A more general problem has tests which each determine the success or failure of some particular subset of the tasks. Any one of the tests may be applied at any point in the manufacturing process and will determine the success or failure of all operations which belong to its test set and which have already been performed. The problem solved in this paper had those test sets restricted to being of the form $\{t_1, t_2, \cdots, t_k\}$, which enabled us to give an efficient optimization algorithm. The best known algorithms for the general problem, however, require an amount of time which may be exponential in the number of tasks, restricting their usefulness to only very small problems.

Another related problem has been considered by T. S. Ellington.[3] In his problem, one has a number of alternative methods for accomplishing each task, the alternatives having different cost and success probabilities, and one would like to select the best method for each task in order to obtain the minimum expected cost. Though Ellington was able to give an efficient algorithm for making this selection, it was based on the assumption that each task is automatically tested immediately after it is applied. If, however, we also allow choice as to test points, the problem becomes more complicated. One cannot merely combine our algorithm with Ellington's, though both are dynamic programming algorithms, because they work from opposite ends of the process sequence. The development of an efficient algorithm to simultaneously select optimal test points and optimal task alternatives remains an open problem which could have substantial practical importance in the design of actual manufacturing processes.

ACKNOWLEDGEMENTS

The author is indebted to D. Slepian, W. B. Joyce, W. M. Boyce, and H. O. Pollak for their invaluable suggestions and encouragement.

APPENDIX A

Optimization Algorithm Example

We illustrate the application of the optimization algorithm with the following small example:

	c_i	p_i	C_i
t_1	10	.9	7
t_2	10	.8	8
t_3	25	.9	4
t_4	20	.8	12
t_5	30	.8	10
t_6	15	.9	12

If no tests are applied, the expected cost of the process is simply the sum of the c_i or 110. If all tests, except the last (which can only increase the total expected cost), are applied, the expected cost of the process is

$$c_1 + C_1 + p_1(c_2 + C_2 + p_2\cdot(c_3 + C_3 + p_3\cdot(c_4 + C_4 + p_4 \cdot(c_5 + C_5 + p_5\cdot(c_6))))) \cong 101.77.$$

We now show the successive computations performed in applying the optimization algorithm.

$K(6, i) = 0$ for all i	no test
$K(5, 5) = \min(10 + .8(15), 15) = 15$	no test
$K(5, 4) = \min(10 + .64(15), 15) = 15$	no test
$K(5, 3) = \min(10 + .576(15), 15) = 15$	no test
$K(5, 2) = \min(10 + .4608(15), 15) = 15$	no test
$K(5, 1) = \min(10 + .41472(15), 15) = 15$	no test
$K(4, 4) = \min(12 + .8(45), 45) = 45$	no test
$K(4, 3) = \min(12 + .72(45), 45) = 44.4$	test
$K(4, 2) = \min(12 + .576(45), 45) = 37.92$	test
$K(4, 1) = \min(12 + .5184(45), 45) = 35.328$	test
$K(3, 3) = \min(4 + .9(65), 64.4) = 62.5$	test
$K(3, 2) = \min(4 + .72(65), 57.92) = 50.8$	test
$K(3, 1) = \min(4 + .648(65), 55.328) = 46.12$	test

$$K(2, 2) = \min (8 + .8(87.5), 75.8) = 75.8 \qquad \text{no test}$$
$$K(2, 1) = \min (8 + .72(87.5), 71.12) = 71 \qquad \text{test}$$
$$K(1, 1) = \min (7 + .9(85.8), 81) = 81 \qquad \text{no test}$$

These all are computed by simply following the optimization algorithm as given, noting by "test" or "no test" whether the first or second value was smaller. The optimal solution has expected cost $c_1 + 81 = 91$, which is appreciably smaller than the two "obvious" solutions. We remind the reader that the cost per fault-free item produced is obtained simply by dividing the cost per item processed by $\prod_{i=1}^{n} p_i$, the probability that the processed item is fault free. Thus, in this case, the optimal solution has cost per fault-free item approximately equal to 233, compared to 282 if no tests are used and 261 if all tests are used. Considerably greater benefits may be derived when the algorithm is applied to processes with a larger number of tasks.

To determine which tests are applied in the optimal solution, one notes first that T_1 is not applied since "no test" was indicated for $K(1, 1)$. Since T_1 is not applied, we go to $K(2, 1)$ which indicates that T_2 should be used. Since T_2 is applied, we go to $K(3, 3)$ which indicates that T_3 should be used. Since T_3 is applied, we go to $K(4, 4)$ which indicates that T_4 should not be applied. Next, since T_4 is not used, we go to $K(5, 4)$ which indicates T_5 is not used. Finally, $K(6, 4)$ indicates that T_6 is not used. Thus, the optimal solution uses only two tests, T_2 and T_3.

If one is required to apply T_6, the optimal solution can be computed similarly to have expected cost 98.6. It uses only tests T_2, T_4, and T_6. In this case, the solution which uses no tests other than T_6 has expected cost 122, and the solution which uses all the tests has expected cost 106.75. We emphasize again that, even though the gain from applying the optimization algorithm to this small example is not insubstantial, considerably greater savings can be obtained with larger, more realistic problems.

APPENDIX B

Proof of Theorem 1

Let

$$D(j, i) = C_i + \left(\prod_{k=i}^{j} p_k\right)(c_{j+1} + K(j + 1, j + 1)) - c_{j+1} - K(j + 1, i).$$

Theorem 1: For all j, $1 < j < n$, and all i, $1 < i \leqq j$, $D(j, i) \geqq D(j, i - 1)$.

Proof. Suppose $D(j, i) < D(j, i-1)$. Then

$$D(j, i-1) - D(j, i) = \left(\prod_{k=i}^{j} p_k\right)(p_{i-1} - 1)(c_{j+1} + K(j+1, j+1)) + K(j+1, i) - K(j+1, i-1) > 0.$$

Then

$$K(j+1, i) - K(j+1, i-1) > \left(\prod_{k=i}^{j} p_k\right)(1 - p_{i-1})(c_{j+1} + K(j+1, j+1)).$$

We shall show that this inequality is never satisfied; that is, for all j, $2 < j \leqq n$, and for all i, $1 < i \leqq j$,

$$(*) \qquad K(j, i) - K(j, i-1) \leqq \left(\prod_{k=i}^{j-1} p_k\right)(1 - p_{i-1})(c_j + K(j, j)).$$

We prove (*) by induction on j, running from n down to 3. If $j = n$, we have that $K(n, i) = K(n, i-1)$, so that the left side of (*) is zero. Since the right side is nonnegative, the inequality is clearly satisfied.

Now suppose that (*) holds for all j, $m < j \leqq n$, and all i, $1 < i \leqq j$. We shall show that it must then hold for $j = m$.

To do this, we first prove a useful intermediate inequality,

$$p_m(c_{m+1} + K(m+1, m+1)) \leqq c_m + K(m, m).$$

We consider two cases. If $D(m, m) < 0$,

$$c_m + K(m, m) = c_m + p_m(c_{m+1} + K(m+1, m+1)) \geqq p_m(c_{m+1} + K(m+1, m+1)).$$

If $D(m, m) \geqq 0$, $c_m + K(m, m) = c_m + c_{m+1} + K(m+1, m)$. By the induction hypothesis, we have

$$K(m+1, m+1) - K(m+1, m) \leqq (1 - p_m)(c_{m+1} + K(m+1, m+1)).$$

Rewriting, we obtain

$$c_{m+1}K(m+1, m) \geqq p_m(c_{m+1} + K(m+1, m+1)),$$

which completes the proof of the intermediate inequality.

We now prove (*) for $j = m$, using two cases.

Case 1. $D(m, i-1) < 0$. Since $D(m, i-1) < 0$,

$$K(m, i-1) = C_m + \left(\prod_{k=i-1}^{m} p_k\right)(c_{m+1} + K(m+1, m+1)).$$

It is always the case that

$$K(m, i) \leqq C_m + \left(\prod_{k=i}^{m} p_k\right)(c_{m+1} + K(m+1, m+1)).$$

Thus,

$$\begin{aligned} K(m, i) - K(m, i-1) &\leqq \left(\prod_{k=i}^{m} p_k\right)(1 - p_{m-1})(c_{m+1} + K(m+1, m+1)) \\ &= \left(\prod_{k=i}^{m-1} p_k\right)(1 - p_{m-1})[p_m(c_{m+1} + K(m+1, m+1))] \\ &\leqq \left(\prod_{k=i}^{m-1} p_k\right)(1 - p_{m-1})[c_m + K(m, m)], \end{aligned}$$

using the intermediate inequality proved earlier. This completes the proof of (*) for Case 1.

Case 2. $D(m, i-1) \geqq 0$. Since $D(m, i-1) \geqq 0$,

$$K(m, i-1) = c_{m+1} + K(m+1, i-1).$$

It is always the case that $K(m, i) \leqq c_{m+1} + K(m+1, i)$. Thus, $K(m, i) - K(m, i-1) \leqq K(m+1, i) - K(m+1, i-1)$. By the induction hypothesis,

$$\begin{aligned} K(m+1, i) - K(m+1, i-1) &\leqq \left(\prod_{k=i}^{m} p_k\right)(1 - p_{i-1})(c_{m+1} + K(m+1, m+1)) \\ &= \left(\prod_{k=i}^{m-1} p_k\right)(1 - p_{i-1})[p_m(c_{m+1} + K(m+1, m+1))] \\ &\leqq \left(\prod_{k=i}^{m-1} p_k\right)(1 - p_{i-1})[c_m + K(m, m)], \end{aligned}$$

again using the previously proved intermediate inequality. This completes the proof of (*) for Case 2.

Theorem 1 follows by induction.

REFERENCES

1. Bellman, R., *Dynamic Programming*, Princeton University Press, Princeton, New Jersey, 1957.
2. Garey, M. R., "Optimal Task Sequencing with Precedence Constraints," to be published.
3. Ellington, T. S., "Cost Optimization Utilizing Reference Techniques (COURT)," Western Electric Engineer, *15*, No. 1 (1971), pp. 25–32.